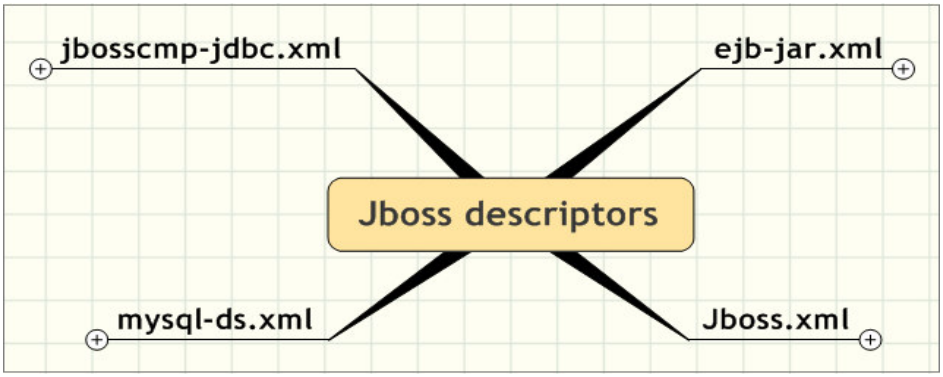
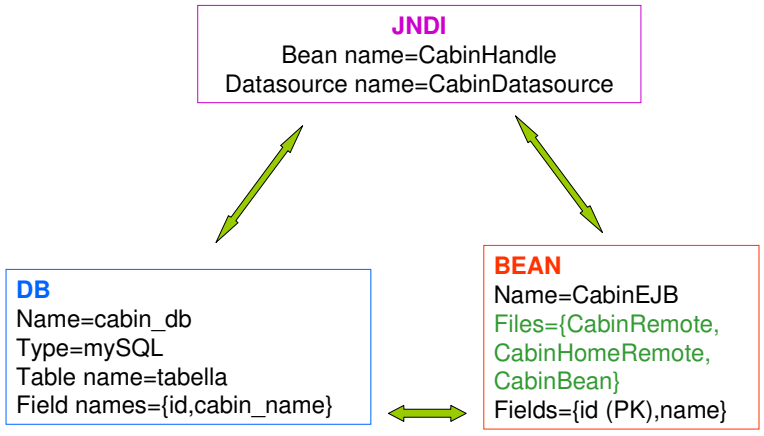
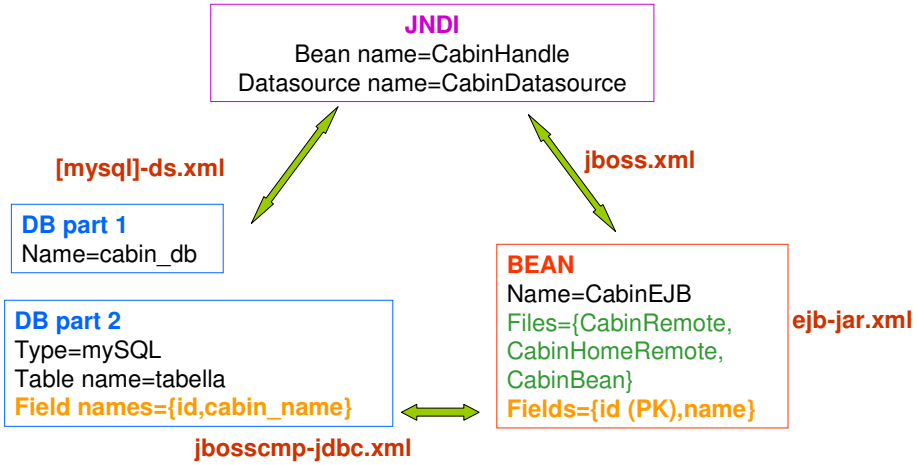
Jboss descriptors



Logical structure



File mapping



ejb-jar.xml

Defines bean name and properties

```
<ejb-jar>
  <enterprise-beans>
    <entity>
      <ejb-name>CabinEJB
      <home>com.titan.cabin.CabinHomeRemote
      <remote>com.titan.cabin.CabinRemote
      <ejb-class>com.titan.cabin.CabinBean
      <persistence-type>Container
      <prim-key-class>java.lang.Integer
      <reentrant>False
      <cmp-version>2.x
      <abstract-schema-name>Cabin
      <cmp-field><field-name>id
      <cmp-field><field-name>name
      <primkey-field>id
      <security-identity><use-caller-identity/>
```

Bean name declaration

Code

Bean Properties

Code

jboss.xml

Binds the bean to a JNDI name

```
<jboss>
    <enterprise-beans>
        <entity>
            <ejb-name>CabinEJB
            <jndi-name>CabinHandle
```

ejb-jar.xml

jndi

```
Object ref = jndiContext.lookup("CabinHandle");
CabinHomeRemote home = (CabinHomeRemote)
        PortableRemoteObject.narrow(ref, CabinHomeRemote.class);
```

mysql-ds.xml

Binds a datasource (DB) to a JNDI name, Specifies some DB properties

```
<datasources>
    <local-tx-datasource>
        <jndi-name>CabinDatasource
        <connection-url> jdbc:mysql://localhost:3306/cabin_db
        <driver-class>org.gjt.mm.mysql.Driver
        <user-name>root
        <password/>
```

jndi

DataBase

Database properties

jbosscmp-jdbc.xml

Specifies the mapping between Bean fields and DB fields Specifies some DB properties

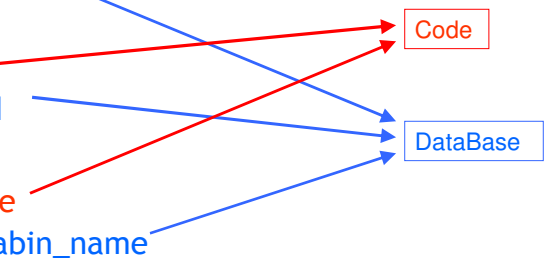
<jbosscmp-jdbc>
  <defaults>
  <enterprise-beans>

<defaults>

<datasource>java:/CabinDatasource
<datasource-mapping>mySQL
<create-table>true
<remove-table>false
<read-only>false
<read-time-out>0
<pk-constraint>true
<read-ahead>
  <strategy>on-load
  <page-size>15
  <eager-load-group>*

jbosscmp-jdbc.xml <enterprise-beans>

```
<entity>
    <ejb-name>CabinEJB
    <datasource>java:/CabinDatasource
    <datasource-mapping>mySQL
    <table-name>tabella
    <cmp-field>
        <field-name>id
        <column-name>id
    <cmp-field>
        <field-name>name
        <column-name>cabin_name
```



Introduction to Entity Beans

An example: part 2-the client

JNDI'role

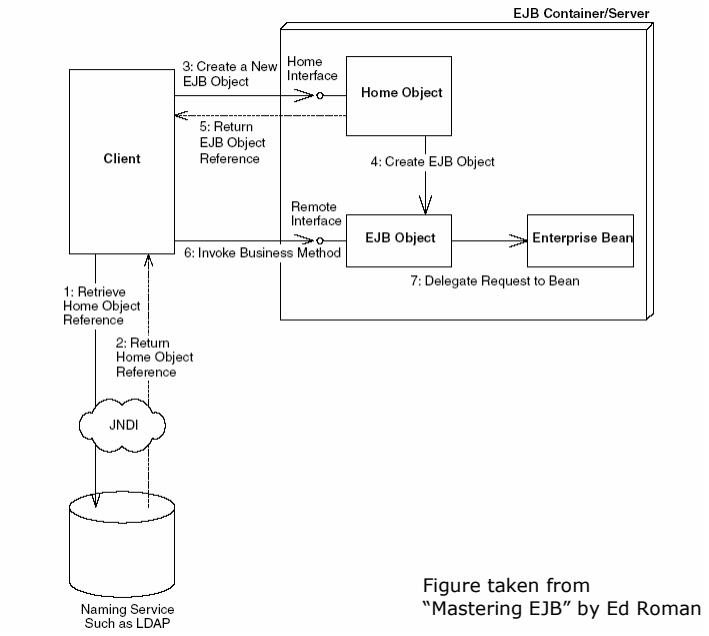


Figure taken from
"Mastering EJB" by Ed Roman

jndi.properties

Il file jndi.properties deve essere nella application root

java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
java.naming.provider.url=localhost:1099

Esempio 1

Client

```
import com.titan.cabin.CabinRemote;
import javax.naming.InitialContext;
import javax.naming.Context;
import javax.rmi.PortableRemoteObject;

public static Context getInitialContext()
    throws javax.naming.NamingException
  {
    return new InitialContext();
 /** context initialized by jndi.properties file
        Properties p = new Properties();
        p.put(Context.INITIAL_CONTEXT_FACTORY,
        "org.jnp.interfaces.NamingContextFactory");
        p.put(Context.URL_PKG_PREFIXES,
        "jboss.naming:org.jnp.interfaces");
        p.put(Context.PROVIDER_URL, "localhost:1099");
        return new javax.naming.InitialContext(p);
    */
  }
}
```

Contesto Inizializzato da Jndi.properties

Una possibile alternativa sarebbe:

Esempio 1

Client

```
public class Client_1
{
  public static void main(String [] args)
  {
    String command[]={
    "create",
    "read",
    "update",
    "delete"
    };
    if (args.length==0) {
      System.out.println("Usage:");
      System.out.println(command[0]);
      System.out.println(command[1]);
      System.out.println(command[2] +" name");
      System.out.println(command[3]);
      System.exit(0);
    }
    // continua...
```

Esempio 1

Client

Acquisici il contesto e stampalo

```
try
    {
     Context jndiContext = getInitialContext();
      java.util.Hashtable ht=jndiContext.getEnvironment();

      System.out.println("INITIAL_CONTEXT_FACTORY
        "+ht.get(Context.INITIAL_CONTEXT_FACTORY));

       System.out.println("URL_PKG_PREFIXES
        "+ht.get(Context.URL_PKG_PREFIXES));

       System.out.println("PROVIDER_URL
        "+ht.get(Context.PROVIDER_URL));

        Iterator i=ht.keySet().iterator();
       while (i.hasNext()) {
         Object o=i.next();
         System.out.println(o+" "+ht.get(o));
       }
```

Esempio 1

Client

```
Object ref = jndiContext.lookup("CabinHandle");
CabinHomeRemote home = (CabinHomeRemote)
   PortableRemoteObject.narrow(ref,CabinHomeRemote.class);

if (args[0].equals(command[0])) { //create
  CabinRemote cabin_1 = home.create(new Integer(1));
  cabin_1.setName("Master Suite")
  cabin_1.setDeckLevel(1);
  cabin_1.setShipId(1);
  cabin_1.setBedCount(3);
 }

 else  if (args[0].equals(command[1])) { //read
  Integer pk = new Integer(1);
  CabinRemote cabin_2 = home.findByPrimaryKe
  System.out.println(cabin_2.getName());
  System.out.println(cabin_2.getDeckLevel());
  System.out.println(cabin_2.getShipId());
  System.out.println(cabin_2.getBedCount());
 }
```

home.create

home.find

Esempio 1

Client

```
else  if (args[0].equals(command[2])) { //update
  Integer pk = new Integer(1);
  CabinRemote cabin_3 = home.findByPrimaryKey(pk);
  cabin_3.setName(args[1]);
  System.out.println("Scritto: "+args[1]);
 }

else  if (args[0].equals(command[3])) { //delete
  Integer pk = new Integer(1);
  CabinRemote cabin_4 = home.findByPrimaryKey(pk);
  cabin_4.remove();
  System.out.println("Cancellato");
 }

else {
  System.out.print("Unrecognized command: "+args[0]);
 }
}
```

home.find

home.find

Esempio 1

Client

```
catch (java.rmi.RemoteException re){re.printStackTrace();}
catch (javax.ejb.RemoveException re){re.printStackTrace();}
catch (javax.naming.NamingException ne){ne.printStackTrace();
catch (javax.ejb.CreateException ce){ce.printStackTrace();}
catch (javax.ejb.FinderException fe){fe.printStackTrace();}
 }
}
```

Tratta le eccezioni e termina

- **Esecuzione:**
  - **ant run.client –Darg1=create** OK
  - **ant run.client –Darg1=read** OK
  - **ant run.client –Darg1=create** ERROR!
  - **ant run.client –Darg1=update pippo** OK
  - **ant run.client –Darg1=read** OK
  - **ant run.client –Darg1=delete** OK
  - **ant run.client –Darg1=read** ERROR!
  - **ant run.client –Darg1=create** OK

Esempio 1

Client

Perche'
Object ref = jndiContext.lookup("CabinHandle");
Invece di
Object ref = jndiContext.lookup("CabinEJB");
?

jboss.xml

```
<?xml version="1.0" ?>
<jboss>
  <enterprise-beans>
    <entity>
      <ejb-name>CabinEJB</ejb-name>
      <jndi-name>CabinHandle</jndi-name>
    </entity>
  </enterprise-beans>
</jboss>
```

Introduction to Entity Beans

An example: part 3-using ant

The build.xml

```
<?xml version="1.0"?>
<!-- JBoss build file                                          -->
<project name="JBoss" default="ejbjar" basedir=".">

  <property environment="env"/>
  <property name="src.dir" value="${basedir}/src/main"/>
  <property name="src.resources"
        value="${basedir}/src/resources"/>
  <property name="jboss.home" value="${env.JBOSS_HOME}"/>
  <property name="build.dir" value="${basedir}/build"/>
  <property name="build.classes.dir" value="${build.dir}/classes"/>

  <!-- Build classpath -->
  <path id="classpath">
      <fileset dir="${jboss.home}/client">
        <include name="**/*.jar"/>
      </fileset>
        <pathelement location="${build.classes.dir}"/>
        <!-- So that we can get jndi.properties for InitialContext -->
        <pathelement location="${basedir}/jndi"/>
  </path>
```

The build.xml

```
<property name="build.classpath" refid="classpath"/>
<!-- Prepares the build directory                              -->
<target name="prepare" >
   <mkdir dir="${build.dir}"/>
   <mkdir dir="${build.classes.dir}"/>
  </target>

<!-- Compiles the source code                                  -->
<target name="compile" depends="prepare">
   <javac srcdir="${src.dir}"
        destdir="${build.classes.dir}"
        debug="on"
        deprecation="on"
        optimize="off"
        includes="**">
         <classpath refid="classpath"/>
   </javac>
  </target>
```

The build.xml

```
<target name="ejbjar" depends="compile">
   <jar jarfile="build/titan.jar">
    <fileset dir="${build.classes.dir}">
          <include name="com/titan/cabin/*.class"/>
    </fileset>
    <fileset dir="${src.resources}/">
       <include name="**/*.xml"/>
    </fileset>
   </jar>
   <copy file="build/titan.jar"
        todir="${jboss.home}/server/default/deploy"/>
 </target>

 <!-- Cleans up generated stuff                                    -->
  <target name="clean.db">
   <delete dir="${jboss.home}/server/default/db/hypersonic"/>
 </target>
 <target name="clean">
   <delete dir="${build.dir}"/>
   <delete file="${jboss.home}/server/default/deploy/titan.jar"/>
 </target>
```

The build.xml

```
 <target name="run.client" depends="ejbjar">
  <java classname="com.titan.clients.Client_1" fork="yes" dir=".">
    <classpath refid="classpath"/>
    <arg value="${arg1}"/>
    <arg value="${arg2}"/>
  </java>
 </target>

</project>
```

NOTA: lanciare con $ant run.client –Darg1=update –Darg2=pippo

Introduction to Entity Beans

More details

Actors:

Component Interface:

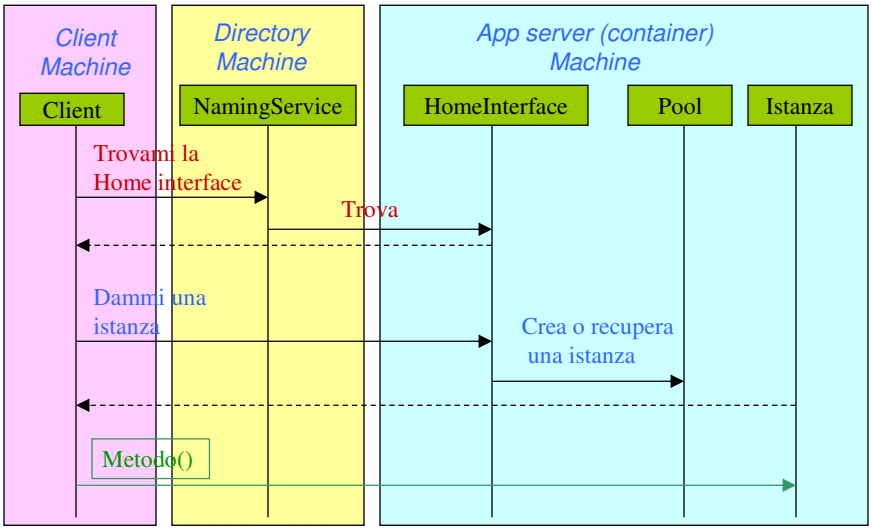
- **Same role as in RMI**

Component Implementation:
Same role as in RMI

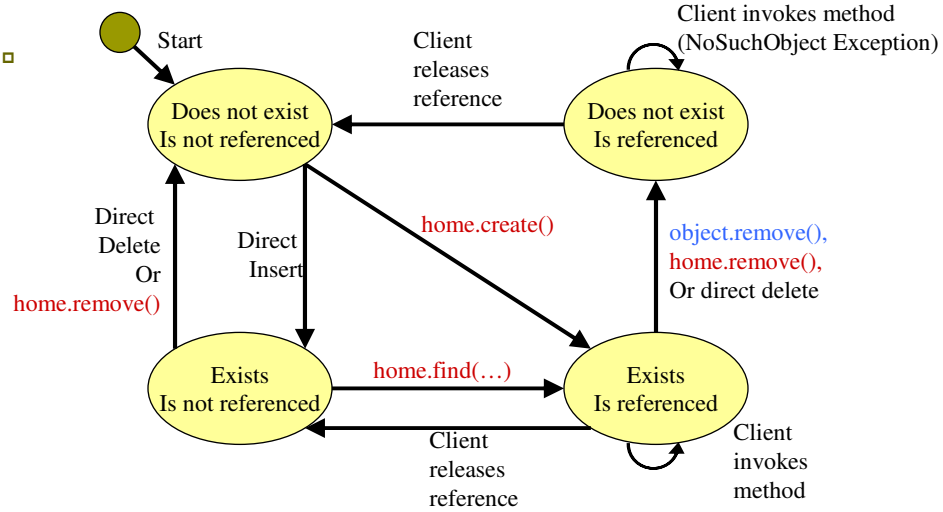
Home Interface:
Implemented by the server, plays the role of a factory class for the component

JNDI service:
Plays the same role as the register in RMI

The logical architecture

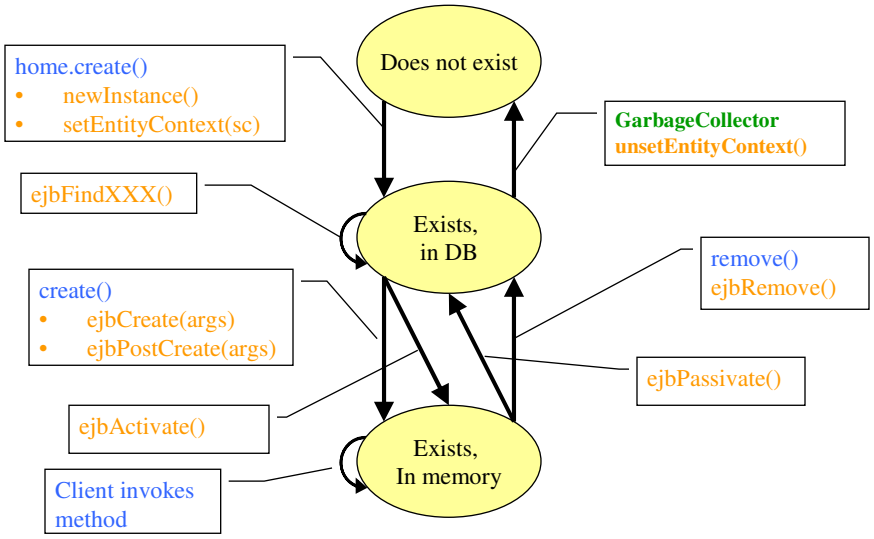


Entity Beans Lifecycle: client's view



14

Entity Beans Lifecycle (stati, transizioni e callbacks)



Introduction to Entity Beans

Local interfaces

Esempio 1

1. CabinLocal

```
package com.titan.cabin;

import java.rmi.EJBException;

public interface CabinLocal extends javax.ejb.EJBLocalObject
{
   public String getName() throws EJBException;
   public void setName(String str) throws EJBException;
   public int getDeckLevel() throws EJBException;
   public void setDeckLevel(int level) throws EJBException;
   public int getShipId() throws EJBException;
   public void setShipId(int sp) throws EJBException;
   public int getBedCount() throws EJBException;
   public void setBedCount(int bc) throws EJBException;
}
```

La classe deve estendere EJBLocalObject

Lancia EJBException anziche' RemoteException

Esempio 1

1. CabinHomeLocal

```
package com.titan.cabin;

import java.rmi.EJBException;
import javax.ejb.CreateException;
import javax.ejb.FinderException;

public interface CabinHomeLocal
                        extends javax.ejb.EJBLocalHome
{
   public CabinLocal create(Integer id)
      throws CreateException, RemoteException;

   public CabinLocal findByPrimaryKey(Integer pk)
      throws FinderException, RemoteException;
}
```

L'interfaccia deve estendere EJBHome

DEFINIRE il metodo create e i finder

ejb-jar.xml

```
<enterprise-beans>
  <entity>
    <ejb-name>CabinEJB</ejb-name>
    <home>com.titan.cabin.CabinHomeRemote</home>
    <remote>com.titan.cabin.CabinRemote</remote>
    <local-home>com.titan.cabin.CabinHomeLocal</local-home>
    <local>com.titan.cabin.CabinLocal</local>
    <ejb-class>com.titan.cabin.CabinBean</ejb-class>
    ...
  </entity>
</enterprise-beans>
```

Esempio 1

1. Client

```
Object ref = jndiContext.lookup("CabinHomeRemote");
CabinHomeRemote home = (CabinHomeRemote)
        PortableRemoteObject.narrow(ref,CabinHomeRemote.class);
...
CabinRemote cabin_1 = home.create(new Integer(1));
...
CabinRemote cabin_2 = home.findByPrimaryKey(pk);
```

```
CabinHomeLocal home = (CabinHomeLocal)
        jndiContext.lookup("java:comp/env/ejb/CabinHomeLocal");
...
CabinLocal cabin_1 = home.create(new Integer(1));
...
CabinLocal cabin_2 = home.findByPrimaryKey(pk);
```

We looked up a bean in *java:comp/env/ejb*.
This is the JNDI location that the EJB specification recommends (but does not require) you put beans that are referenced from other beans.